WEST

Generate Collection Print

L8: Entry 3 of 4 File: USPT Aug 19, 1997

DOCUMENT-IDENTIFIER: US 5659737 A
TITLE: Methods and apparatus for data compression that preserves order by using failure greater than and failure less than tokens

Abstract Paragraph Left (1):
A compressor receives a plurality of characters in a character string, and encodes the input character string to generate an encoded string. The encoding preserves the original binary order of the input character strings in the encoded strings. A predicted character is generated for each character based on prior character sequences in the character string. In one embodiment, a hash based predictive technique is used to generate the predicted characters. A correct order indicating token is generated for an input character if a predicted character, corresponding to the input character, is the input character. The compressor stores the order indicating token in the encoded string to represent the character. However, if an input character has a binary representation greater than a binary representation of a corresponding predicted character, then a failure greater than token and the input character are stored in the encoded string to represent the input character. A failure less than token and an input character are stored for an input character in the encoded string if the input character has a binary representation less than a binary representation of the predicted character. Thus, binary order is preserved in the encoded string. With the binary order preserving compression technique, data may be sorted while the data are still compressed. Prefix bits compression and dynamic token generation are also disclosed.

Brief Summary Paragraph Right (2):
Generally, data compression is used in systems to reduce the amount of storage space required to store the data or to reduce the amount of bandwidth required to transmit the data. Various data compression techniques are known in the prior art. For example, hash-based predictive compression is used to compress character strings of natural language text. Hash-based predictive compression utilizes the fact that the knowledge of a short substring of characters constitutes a good basis for predicting the next character in the character string. This method for predicting successive characters based on a preceding substring is feasible because the order of characters in a natural language is not random.

Brief Summary Paragraph Right (3):
In general, the hash based predictive compression technique gathers the information of short substrings of characters, and predicts the next character in the sequence based on the substring. For a correct prediction, the compression technique does not store the corresponding correctly predicted character. Instead, an indication is stored to reflect that the character was predicted correctly. However, by eliminating the storage of some characters and adding indications as to whether the character is stored, the binary representation of the characters no longer exhibits the original binary order. For example, the letter "B" has a binary representation greater than the binary representation of the letter "A." However, when encoded, the letter "B" may have a binary representation less than the binary representation of the letter "A."

Brief Summary Paragraph Right (6):
A plurality of characters in a character string are input to a compressor. The compressor encodes the input character string to generate an encoded string. The encoding preserves the original binary order of the character strings in the encoded

strings. A predicted character is generated for each character based on prior character sequences in the character string. In one embodiment, a hash based predictive technique is used to generate the predicted characters. A correct order indicating token is generated for an input character if a predicted character, corresponding to the input character, is the input character. The compressor encodes the order indicating token in the encoded string to represent the character. However, if an input character has a binary representation greater than a binary representation of a corresponding predicted character, then a failure greater than token and the input character are encoded to represent the input character. A failure less than token and an input character are encoded for an input character if the input character has a binary representation less than a binary representation of the predicted character. When an input character is stored with a token, the tokens are appended in the most significant bit location so that the correct order indicating token, the failure greater than token, and the failure less than token preserves the original binary order of the character string.

Drawing Description Paragraph Right (5):
FIG. 3 is a flow diagram illustrating one embodiment of hash based predictive compression that preserves binary order.

Drawing Description Paragraph Right (8):
FIG. 6 is a flow diagram illustrating one embodiment for compressing prefix bits when using a hash based predictive compression technique that preserves binary order.

Detailed Description Paragraph Right (2):
The compressor 100 is coupled to a hash table 110. The hash table 110 contains a plurality of entries that store predicted characters. The predicted characters correspond to a successor character in a given substring of characters. The compressor 100 generates a hash index and receives a predicted character from the hash table identified by the hash index. The hash index is derived from a hash function. In general, a hash based predictive compression technique utilizes statistical information about a character string to derive the entries in the hash table.

Detailed Description Paragraph Right (3):
The present invention is described in conjunction with a hash based predictive compression technique that utilizes a given hash table; however, any method used to generate the predicted values for the hash based predictive compression technique may be used in conjunction with the present invention without deviating from the spirit and scope of the present invention.

Detailed Description Paragraph Right (4):
The binary order preserving data compression technique of the present invention is described in conjunction with a specific hash-based predictive compression algorithm. The hash based predictive compression technique disclosed may be used in conjunction with the present invention to encode a character string or textual data. However, this hash based predictive compression technique is merely exemplary, and any hash based algorithm may be used in conjunction with the binary order preserving techniques disclosed herein without deviating from the spirit and scope of the invention.

Detailed Description Paragraph Right (8):
FIG. 3 is a flow diagram illustrating one embodiment of hash based predictive compression that preserves binary order. In block 300, the hash table 110 is initialized to a given state. As shown in block 310, an initial block of characters for the character string are stored in the encoded string 115 (e.g. the first "k" characters). For all subsequent characters, (C.sub.i +k+1), the compressor 100 calculates a hash index or hash address based on the hash function used as shown in block 320. With the hash address, the hash table 110 is indexed to retrieve the predicted character (P.sub.i). A comparison is made between the current character (C.sub.i) and the predicted character (P.sub.i) to determine whether the current character, C.sub.i, is equal to the predicted character, P.sub.i, as shown in block 340. If not, the compressor 100 generates a failure order indicating token depending upon whether the binary representation of the predicted character is greater or less

than the binary representation of the current character as shown in block 360. The failure order indicating token and the current character, Ci, are stored in the encoded string 115 in the "i" location. The failure order indicating token, appended to the beginning of the character, Ci, preserves binary order relative to the predicted character. If the predicted character is the same as the current character, then the compressor 100 generates a correct order indicating token that preserves binary order relative to the greater than or less than token as shown in block 350. If the next character in the input character string 105 is not the end of the character string, then the compressor 100 processes the next character Ci in accordance with blocks 320, 330, 340, 350, 360, and 370.

Detailed Description Paragraph Right (9):
FIG. 1b is a block diagram illustrating one embodiment of decompression. A decompressor 120 receives, as input, the encoded string 115, and generates, as an output, the character string 105. As shown in FIG. 1b, the decompressor 120 is coupled to the hash table 110 such that the decompressor 120 accesses the hash table 110 with the hash index to receive a predicted character. FIG. 4 is a flow diagram illustrating one embodiment for decompressing the encoded string. The hash table 110 is initialized to the same state as it was initialized for compression as shown in block 400. As shown in block 410, the decompressor 120 retrieves the initial "k" characters from the encoded string 115, and places these characters at the beginning of the character string 105.

Detailed Description Paragraph Right (10):
The decompressor 120 processes each entry in the encoded string 115 (i.e. each token and corresponding character, if any). The hash index or address is calculated from the hash function, and is utilized to retrieve the predicted character for the corresponding input from the encoded string as shown in block 420. The decompressor 120 reads the token, and if the token is a correct order indicating token, then the character string entry, C.sub.i, is set to the predicted character P.sub.i as shown in block 440. If the character was improperly predicted, then the decompressor 120 sets the character C.sub.i to the value contained in the encoded string 115 as shown in block 450. As shown in FIG. 4, for each input E.sub.i, the decompressor 120 executes blocks 430, 440, and 450.

Detailed Description Paragraph Right (13):
FIG. 6 is a flow diagram illustrating one embodiment for compressing prefix bits when using a hash based predictive compression technique that preserves binary order. In general, the prefix bits compression technique reduces the number of bits contained in the encoded string. The bits that are eliminated provide redundant information in terms of the binary order. Therefore, the order of the binary representation of the encoded characters remains preserved. As shown in blocks 600, 610 and 620, if the current character is less than the predicted character, and the predicted character contains "n" zeros in the most significant bit position, then the "n" most significant bits in the current character are eliminated for output to the encoded string. The "n" most significant bits may comprise one or more bits. As shown in blocks 600, 630, and 640, if the current character is greater than the predicted character, and there are "n" most significant bits equal to "1" in the predicted character, then the "n" most significant bits in the current character are eliminated for storage in the encoded string. Again, the "n" most significant bits may comprise one or more bits. Through use of prefix bits compression, a reduction of bits in the encoded string is obtained without sacrificing loss of binary order.

Detailed Description Paragraph Left (2):
where h is the hash function. The hash table stores the k parameter block size as well for future reference during decompression.

Detailed Description Paragraph Left (3):
where the devisor M determines the effective size of the hash table and is a prime number. The key is defined as the predictive substring. The remainder of division hash function works well when the block size is relatively small, such as when the block size is 3.